# Robust Segmentation of Unconstrained Online Handwritten Documents

Anoop M. Namboodiri and Anil K. Jain
Department of Computer Science and Engineering
Michigan State University, East Lansing, MI - 48823
{anoop,jain}@cse.msu.edu

## Abstract

A segmentation algorithm, which can detect different regions of a handwritten document such as text lines, tables and sketches will be extremely useful in a variety of applications such as retrieval, translation and genre classification. However, this task is extremely challenging for handwritten documents, which vary considerably in their structure and content. In this paper, we describe a robust segmentation method to detect the regions in an unstructured on-line handwritten document. We utilize the temporal information in on-line documents along with its spatial layout to improve the segmentation results. The properties of handwritten strokes are computed using a spline-based representation. We compute the most likely segmentation of the handwritten page using a Stochastic Context Free Grammar based parser. The regions considered in this work include paragraphs, text lines, words, and non-text regions.

## 1. Introduction

With advances in hardware technology, a variety of devices have now become popular, which allows the on-line capture of a user's handwriting (e.g., Tablet PC, IBM ThinkPad TransNote, Ink Link, Anoto Pen [11]). These devices let users take notes on their computers using an interface (pen), which is both familiar to them and is less distracting than a keyboard. However, the notes generated using such an interface is often highly unstructured in its layout. The problem of segmentation of such a handwritten page into its component regions is of great interest in a variety of applications. Understanding the layout of a handwritten page leads to more reliable text recognition, and helps in preserving the spatial relationship of different regions in a page (such as a figure and its caption). In addition, the layout of a page can be used as part of a query in retrieval applications (e.g., to retrieve pages with a sketch on the bottom right). The process of segmentation of text regions into words and lines can help in the identification of structures like unruled tables and lists, which helps a recognizer to format its output, consistent with the input page.

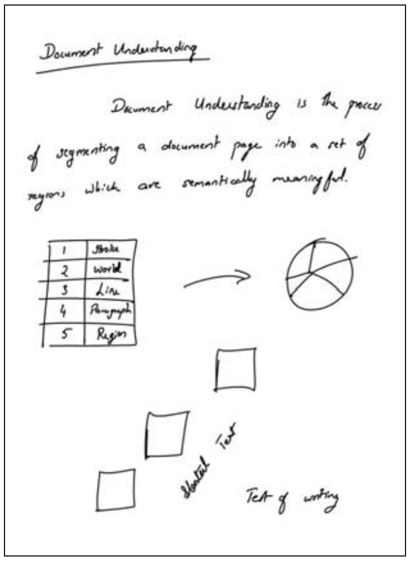
However, the reliable identification of semantically meaningful regions in an unstructured handwritten page is an extremely challenging problem. The difficulty in segmentation mainly arises from two different sources: 1) The inherent ambiguity in region boundaries in handwritten notes and 2) The temporal interleaving of strokes in multiple regions during the handwriting process. A robust system to segment unconstrained handwritten notes has to take these characteristics into consideration, while defining and detecting segment boundaries.

Most of the research in document analysis has focused on off-line (scanned) documents. Examples of this work include page decomposition [5], locating embedded text in color images [14], skew detection [13] and table identification [6, 3, 12, 2, 7]. The work in on-line document analysis till date includes segmentation of text lines [9, 1, 8] and rule-based methods in on-line document segmentation [4, 10]. In this paper, we present a principled approach to deal with the problem of segmentation of on-line handwritten documents.

### 1.1. On-line Data

There are a few important aspects of on-line documents that enable us to process them in a fundamentally different way than off-line documents. The most important characteristic of on-line documents is that they capture the temporal sequence of strokes[1] while writing the document. This allows us to analyze the individual strokes and use the additional temporal information for both document segmentation as well as text detection. However, the temporal information across strokes could also be misleading for segmentation in cases, where the user switches between regions during writing. Figure 1 shows an example of an on-line handwritten page. In addition to the spatial information represented in the figure, the document contains temporal information within strokes and the stroke order. The temporal data is used in defining the segment boundaries along with the spatial layout of the strokes in section 4. The data generated by an on-line data capturing device is a sequence of points,

[1] A stroke is defined as the locus of the tip of the pen from pen-down to the next pen-up position.



Document Understanding

Document Understanding is the process of segmenting a document page into a set of regions which are semantically meaningful.

|   |   |
|---|---|
| 1 | Stroke |
| 2 | Word |
| 3 | Line |
| 4 | Paragraph |
| 5 | Region |

Skatal Text

Text of writing

Figure 1: An on-line handwritten document. In addition to the spatial information, the on-line document contains the temporal information of strokes and the stroke order.

$(x_k, y_k)$, which is obtained by sampling the curves at regular intervals of time. A stroke, $\xi_i$, is hence defined as:

$$\xi_i(k) = < (x_k, y_k) >; \quad k = 0, 1, \cdots, n_i \text{ and } x_k, y_k \in [0, max_{xy}], \qquad (1)$$

where $max_{xy}$ is the maximum value that $x_i$ or $y_i$ can take and $n_i$ is the number of points in stroke $i$. We define a stroke model (section 3), which estimates the most likely trace of the pen-tip, given the noisy data from the sensor.

## 2. Document Segmentation

The process of document segmentation in on-line documents can be defined as partitioning a set of $n$ strokes (representing a page) into $k$ subsets, $\{R_1, R_2, \cdots, R_k\}$, where $R_i$ represents a region and $|R_1| + |R_2| + \cdots + |R_k| = n$. In this paper, we develop a Stochastic Context Free Grammar (SCFG) based solution, which computes an optimal partition of the strokes, given a criterion function. The method assumes that the strokes of a region are contiguous in time and spatially compact. We will define the compactness more precisely in section 4, which forms part of the criterion function. The assumption of temporal contiguity leads to over-segmentation. A post-processing stage is used to combine such fragments into a single region. An overview of the segmentation algorithm is given in figure 2.

The first step is to represent the individual strokes using a stroke model, which helps in removal of noise in the data as well as computation of stroke properties. These properties

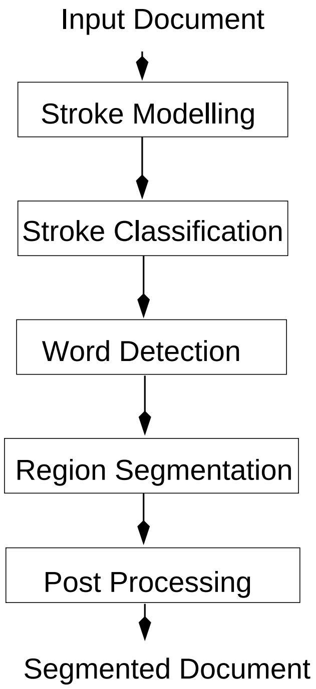


Figure 2: Steps of segmentation in the proposed algorithm.

are then used to classify the individual strokes as text or non-text. The classification retains the uncertainty in the decision process, which is used by the subsequent stages in defining word and region boundaries. Each of the above steps are described in detail in the following sections.

## 3. Stroke Modelling and Classification

The issue of representation of the strokes for the computation of its properties is often overlooked and leads to errors in later stages of segmentation. In this section, we describe a stroke model, which makes the least assumptions about its nature, while helping in removal of noise from the sensed data. In order to decide on a model for representing stroke, we make use of the properties of the handwriting process. We consider each stroke as two independent sequences, $\xi_x(t)$ and $\xi_y(t)$ of $x$ and $y$ positions, both functions of time. Since the force applied on the pen by the fingers is finite, the curvature of the above functions are limited to a maximum value. In other words, the functions $\xi_x(t)$ and $\xi_y(t)$ are 'smooth'. In addition, we assume that the noise in the data has a zero-mean normal distribution (white noise).

Based on these assumptions, we select the cubic spline model to represent the strokes. Splines impose the smoothness constraint, which minimizes the integral of the squared second derivative along the curve. i.e.,

$$\int_a^b S''(t)^2 dt \;\; \leq \;\; \int_a^b G''(t)^2 dt,$$

where $S(t)$ is the spline function and $G(t)$ is the set of all functions, which have continuous second derivatives and $a$ and $b$ are the end points of the curve. We use the B-spline representation in this work to model handwritten strokes.

In order to model handwritten data using B-splines, we need to determine the following parameters:

1. Order of the basis functions
2. The number and position of control points, and
3. the position of knot vectors.

The choice of order of the basis functions is the result of continuity constraints placed on the derivatives. The lowest order basis which satisfies the smoothness constraint is $3$, which forms the cubic b-spline basis. The control points in our application are given directly by the handwriting digitizer, which provides a sequence of $x$ and $y$ positions of the pen tip. The choice of knot vectors will define the nature of the resulting spline. We select an open uniform knot vector sequence, where the distance between two knot vectors is kept constant. This ensures that all parts of the stroke are given equal importance (in absence of evidence to the contrary).

The only parameter that needs to be selected is the inter-knot distance, which will decide the smoothness of the curve. Note that all parameters till now are selected based on the properties of handwritten strokes and are not dependent on any specific type of writing or digitizing device. However, the inter-knot distance needs to be selected based on the sampling rate of the pen tip by the digitizer. In our experiments with the Tablet PC, with a sampling rate of $128$ samples per second, selecting every third sample as a knot vector seems to give good approximation. Figure 3 shows four examples of spline approximations to handwritten strokes. The average error of the approximation function (spline curve) on a typical document was about $1.6\%$ of the stroke size (height).

## 3.1. Computing Stroke properties

Once the strokes are approximated using a spline, we can reliably compute different properties of the stroke. The first step in our algorithm (see figure 2) classifies the strokes into two categories, *text* and *non-text*. The classification requires two properties to be computed from each stroke; the stroke length and the stroke curvature [4]. The stroke length is the total length of the stroke and the stroke curvature is the sum of the absolute differences from linearity of the points along the stroke.

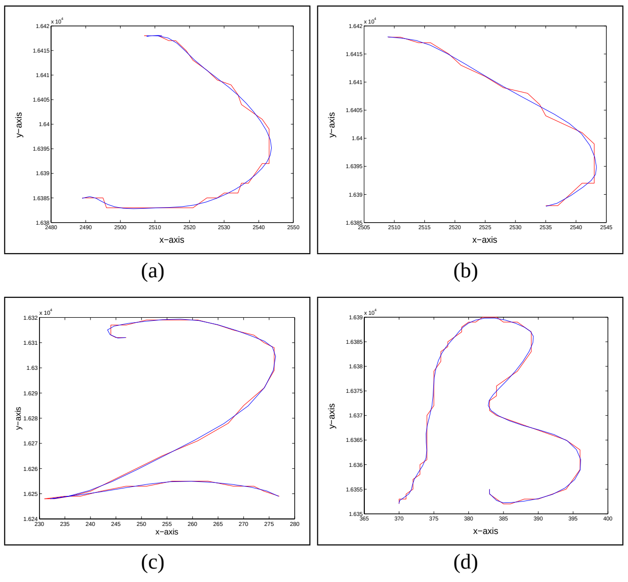


Figure 3: Examples of spline approximation of four strokes.

Let the stroke be defined as $S(t) = (S_x(t), S_y(t))$. The length of the stroke between two points, $t_i$ and $t_j$ is the arc-length integral of the stroke, given by:

$$L(t_i, t_j) = \int_{t_i}^{t_j} ((S_x^{'}(t))^2 + (S_y^{'}(t))^2)^{1/2} \;\, dt.$$

However, there is no closed form solution for the above integral. A numerical solution is often adopted to compute this integral. In the case of handwriting, we can approximate the curve between any two sample points using a straight line due to the inverse relationship between writing speed and curvature. With this assumption, we can convert the integral into the following sum.

$$\sum_{i=1}^{n} L_1(t_i, t_{i+1}),$$

where $L_1$ is the euclidean distance between points $t_i$ and $t_{i+1}$.

The the curvature of the stroke can be computed from the tangent directions, $\Theta(t_i)$, at each point. The total curvature is given by the limiting sum:

$$C(t_i, t_j) = \lim_{\delta t \to 0} \sum_{k=0}^{(t_j - t_i)/\delta_t} |\Theta_{t_i + (k+1)dt} - \Theta_{t_i + k.dt}|,$$

where $\Theta(t)$ is given by:

$$\Theta(t) = \arctan(S_y'(t)/S_x'(t)).$$

However we can approximate (exactly under certain conditions) the limit by the sum:

$$C(t_i, t_j) = \sum_{k=i}^{j-1} |\Theta_{t_{k+1}} - \Theta_{t_k}|.$$

The derivative of a spline function of order $n$ is another spline of order $n-1$. The derivative of $S(t)$ is given by:

$$\frac{d}{dt}S(t) = S'(t) = \sum_{i=0}^{n-1} N_{i,p-1}(t)K_i,$$

where $N_{i,p}(t)$ is the spline basis function, and $K_i$ is given by:

$$K_i = \frac{p}{u_{i+p+1} - u_{i+1}}(P_{i+1} - P_i)$$

Once the length and features are computed, we construct a stroke classifier, which returns the estimated class of each stroke, with a confidence measure. Figure 4 shows a set of text and non-text strokes in the two-dimensional feature space.

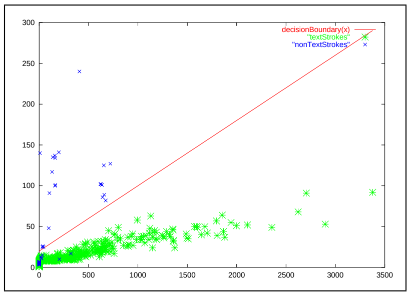


Figure 4: Strokes of text and non-text class in the two-dimensional feature space. The horizontal axis is the stroke curvature and the vertical axis is stroke length.

## 4. Word and Region Segmentation

Once the strokes are classified into text and non-text strokes, we group adjacent text strokes into words. The word detection module is used to identify tokens, which form the terminals of the SCFG, which is used to describe the page layout. One could define the grammar rules to generate the strokes directly. However, such an approach will increase the complexity of the grammar and will affect the parsing performance.

The word detection module itself is a deterministic finite state automaton (FSA), which is augmented to output a confidence value to each word detected. Figure 5 shows the structure of the FSA. The node $S_2$ denotes the end state, where a word is detected. Along with the word detection, we compute a confidence value for the assigned label. This is computed as the posterior probability of the label, given a word model. The probability of observation for the text class is the product of the probabilities of the individual strokes being text and the probability of orientation of the stroke centers.

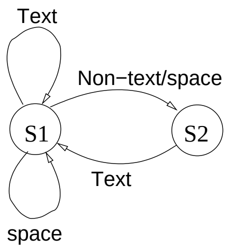


Figure 5: The finite state automaton, which detects words in the document.

$$P(word/strokes) = \prod_{i=1}^{k} P(text/stroke_i) * P(position_i),$$

where $P(position)$ is a normal distribution on the inter-stroke distance with zero mean and variance set based on the resolution of the tablet. A similar value is computed for non-text stroke. Once the conditional class probabilities are estimated, the confidence value for the label is computed using the bayes rule:

$$P(text/word) = \frac{p(word/text)P(text)}{p(word/text) + p(word/nontext)}.$$

We assume equal priors ($P(text) = P(nontext)$) for the labels. However, one could modify this based on the neighboring regions and their spatial relationships. Figure 6 shows example of word detection for the on-line document in figure 1.

### 4.1. Region Segmentation

The segmentation of regions is inherently an ambiguous problem. Even human experts could differ in their judgement about region boundaries in an unstructured handwritten note. Hence we need to define the criteria, which are generally accepted as defining distinct regions. The criterion which we use to define a region is its spatial and temporal compactness. In other words, a region is a set of strokes that are spatially and temporally close. We define the penalty of clustering a set of strokes into a single region based on the difference in area of its bounding box and that of the individual strokes. However the compactness criterion by itself tend to fragment regions to reduce the total area. Hence we introduce an opposing criterion, which penalizes splitting of regions. Hence the probability associated with a particular partition is inversely proportional to both the number of regions and the sum of the

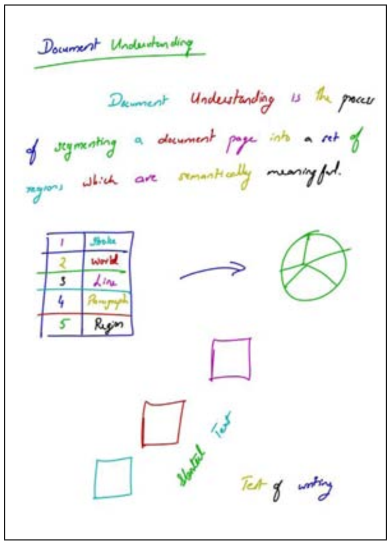

Document Understanding
Document Understanding is the process
of segmenting a document page into a set of
regions which are semantically meaningful.
1 Stroke
2 Word
3 Line
4 Paragraph
5 Region
Slanted Text
Text of writing

Figure 6: Results of word detection. Each word is shown in a different color.

areas enclosed by the bounding boxes of the individual regions. Hence the criterion or cost function that we want to minimize, C(S), for the segmentation $S = \{s_1, s_2, \cdots, s_n\}$, is:

$$C(S) = \frac{area(S)}{n^k . \sum_{i=1}^{n} area(s_i)} . \prod_{i=1}^{n} C(s_i),$$

where the $s_i$ denote the individuals regions. The cost of the terminal symbols are set to 1. $k$ is a constant, which controls the penalty for a division into $n$ regions, and is set experimentally.

The goal of our system is to come up with a partition of the strokes that globally minimizes the cost function. We define a stochastic context-free grammar (SCFG), which helps in parsing the document into a hierarchical segmentation of regions. Since the intermediate nodes in this tree refer to regions of text or sketches, the nodes should carry additional information about the nature and content of the region. Such a grammar, which incorporates attributes of nodes in addition to their labels is referred to as *attributed grammar*. In our system, the attributes include the bounding box of the region and the classification confidence of the region. Figure 7 shows the grammar used in this work for region segmentation. In addition to the grammar rules, each production is associated with a probability. The probabilities for all productions, except that to the text line are computed based on the cost function, $C(S)$ described above. The probability for text lines also incorporates the direction between adjacent words. Note that some of the rules are not listed here to improve readability.

The parsing is done using the CYK algorithm, which is a

| | | |
|---|---|---|
| page | ::= | regions [numberOfRegions] |
| region | ::= | paragraph \| nonTextRegion |
| | | [boundingBox] |
| paragraph | ::= | textLines [separation] |
| textLine | ::= | words [direction] |
| nonTextRegion | ::= | words \| nonTextStrokes |
| | | [boundingBox] |

Figure 7: Grammar rules used for segmentation. The attributes are shown in square brackets.

dynamic programming method to compute the most probable parse of an observation sequence, given a grammar. The result of parsing the page shown in figure 6 is given in figure 8.

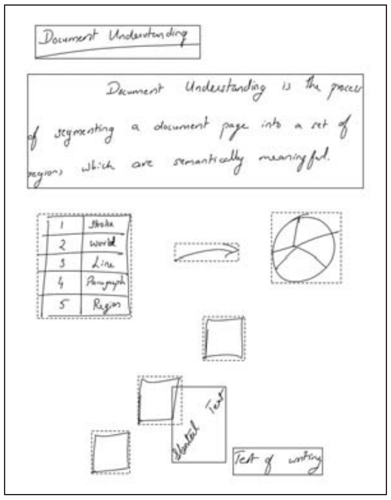
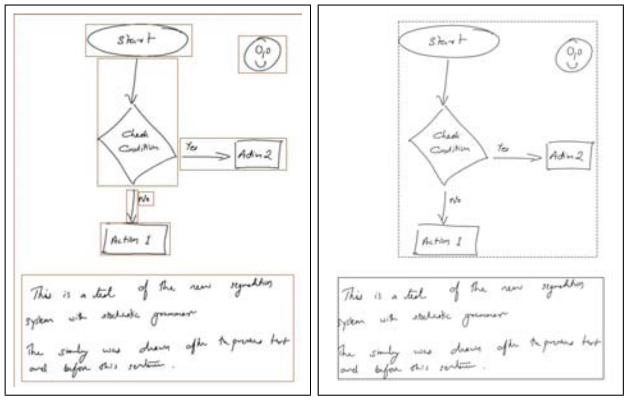

Document Understanding
Document Understanding is the process
of segmenting a document page into a set of
regions which are semantically meaningful.
1 Stroke
2 Word
3 Line
4 Paragraph
5 Region
Slanted Text
Text of writing

Figure 8: Result of parsing of the document in figure 6.

However the parsing might lead to over-segmentation in some cases, where dividing a region could reduce the area of its bounding box. Figure 9(a) shows an example of a document, which is over-segmented by the parsing algorithm. This could also arise from temporal discontinuities in a region. To overcome these difficulties, we introduce a post processing stage, which combines spatially overlapping regions. We also combine spatially and temporally adjacent non-text regions, based on a threshold on spatial adjacency (set to $120$ units for the resolution of Tablet PC). The result of post processing of the document in figure 9(a) is given in figure 9(b). Note that the fragmented flowchart is combined into a single region. However, the sketch at the top left corner is also included in the region, which could be labelled as an error.

The algorithm was tested on a set of $15$ documents containing $80$ regions. To compute the accuracy of segmentation, we define the number of errors in a segmentation as the number of merges or splits required to convert the result



(a) (b)

Figure 9: (a) segmentation and (b) post-processing results of an on-line handwritten document.

into the ground truth. There were a total of $8$ errors in the test set.

## 5. Conclusions and Future Work

We have proposed a framework for robust segmentation of unstructured handwritten documents. The stroke properties are computed using a spline model, which approximates the observed data. A stochastic context free grammar-based solution is proposed for region segmentation. The results are promising and the errors, when they occur, are easy to correct for users. A quantitative comparison of the results to those reported in the literature is not feasible due to the lack of standardization in the test databases and error reporting. However, the proposed approach is capable of segmentation of a variety of writing styles and has the potential of learning from observed data, unlike rule-based methods.

As part of the future work, we will try to improve the parsing results by learning the parameters of the grammar from a set of labelled training samples. Another approach that might improve the result would be to define the grammar to have the strokes directly as terminals. This approach will avoid the word detection module, which is designed based on domain knowledge rather than observed data. Once such a system is set up, one could also include further properties of the stroke to enhance the ability of the parser to define the nature of regions. In addition, we will conduct experiments on a larger database of documents to establish the effectiveness of the approach.

## References

[1] E. Bruzzone and M. Coffetti. An algorithm for extracting cursive text lines. In *Proceedings of the* $5^{th}$ *International Conference on Document Analysis and Recognition (ICDAR'99)*, pages 749–752, Bangalore, India, September 1999.

[2] Y. Hirayama. A method for table structure analysis using DP matching. In *Proceedings of the* $3^{rd}$ *International Conference on Document Analysis and Recognition (ICDAR'95)*, pages 583–586, Montreal, Canada, August 1995.

[3] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table detection across multiple media. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications*, Bangalore, India, September 1999.

[4] A. K. Jain, A. M. Namboodiri, and J. Subrahmonia. Structure in on-line documents. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 844–848, Seattle, Washington, September 2001.

[5] A. K. Jain and B. Yu. Document representation and its application to page decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):294–308, March 1998.

[6] T. G. Keininger and A. Dengel. The T-RECS approach for table structure recognition and table border determination. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications*, Bangalore, India, September 1999.

[7] W. Kornfeld and J. Wattecamps. Automatically locating, extracting and analyzing tabular data. In *Proceedings of* $21st$ *Annual International ACM SIGIR Conference*, pages 347–349, Melbourne, Australia, August 1998.

[8] Y. Pu and Z. Shi. A natural learning algorithm based on Hough transform for text lines extraction in handwritten documents. In *Proceedings of the* $6^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 637–646, Taejon, Korea, August 1998.

[9] E. H. Ratzlaff. Inter-line distance estimation and text line extraction for unconstrained online handwriting. In *Proceedings of the* $7^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, Nijmegen, Netherlands, September 2000.

[10] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 60–65, Edinburgh, UK, August 2003.

[11] J. Subrahmonia. Pen computing: Challenges and applications. In *Proceedings of the* $15^{th}$ *International Conference on Pattern Recognition*, volume 2, pages 60–66, Barcelona, Spain, September 2000.

[12] T. A. Tokuyasu and P. A. Chou. An iterative decoding approach to document image analysis. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications*, Bangalore, India, September 1999.

[13] B. Yu and A. K. Jain. A robust and fast skew detection algorithm for generic documents. *Pattern Recognition*, 29(10):1599–1630, October 1996.

[14] Y. Zhong, K. Karu, and A. K. Jain. Locating text in complex color images. *Pattern Recognition*, 28(10):1523–1535, 1995.